AMENDMENTS TO THE SPECIFICATION:

Page 1, with respect to the amendment preliminary filed October 5, 2005:

This application is the US national phase of international application **PCT/EP2004/003731** filed **7 April 2004**, which designated the U.S. and claims priority to **EP 03290894.9** filed **9 April 2003** and **USSN 10/665,352** filed **22 September 2003** (now U.S. Patent No. 7,434,224), the entire contents of each of which are hereby incorporated by reference. This application is also related to copending commonly owned application Serial Nos. 10/566,274 filed June 18, 2004; 10/573,881 filed September 30, 2004; and 10/573,918 filed October 1, 2004.

Pages 1-2, bridging paragraph:

Applications programs interact with the computers on which they run through operating systems. By using the applications programming interface (API) of the operating system, the applications program can be written in a portable fashion, so that it can execute on different computers with different hardware resources. Additionally, common operating systems such as Linux or Windows provide multi-tasking; in other words, they allow several programs to operate concurrently. To do so, they provide scheduling; in other words, they share the usage of the resources of the computer between the different programs, allocating time to each in accordance with a scheduling algorithm. Operating systems of the this kind are very widely used, but they generally

make no provision for running real time applications, and they therefore are unsuitable for many control or communications tasks.

Page 2, 1st and 2nd full paragraphs:

For such tasks, therefore, real time operating systems have been developed; one example is ChorusOS (also know as Chorus) and its derivatives. Chorus is available as open source software from the World Wide Web:

~~http://www.~~experimentalstuff.com/Technologies/ChorusOS/index.html

and Jaluna at

~~http://www.~~jaluna.com/.

It is described in "ChorusOS Features and Architecture overview" Francois Armand, Sun Technical Report, August 2001, 222p, available from:

~~http://www.~~jaluna.com/developer/papers/COSDESPERF.pdf.

Page 4, 2nd paragraph:

Another approach is that of ADEOS (Adaptive Domain Environment for Operating Systems), described in a White Paper at

~~http://~~opersys.com/ftp/pub/Adeos/adeos.pdf.

Page 4, 4th paragraph:

~~http://www.~~aero.polimi.it/~rtai/applications/.

Pages 8-9, bridging paragraph:

A computer system to which the system is applicable 100 comprises a central processing unit (CPU) 102, such as a Pentium 4™ CPU available from Intel Corporation, or PowerPC CPU available from Motorola (the embodiment has been implemented on both), coupled via a system bus 104 (comprising control, data and address buses) to a read-only memory (ROM) chip 106; one or more banks of random access memory (RAM) chips (108); disk controller devices 110 (for example, integrated development environment (IDE) or small computer system interface (SCSI) controllers, connected to a floppy disk drive, a hard disk drive, and additional removable media drives such as digital video disk (DVD) drives[[)]]; one or more input/output ports (112) (for example, one or more universal serial bus (USB) port controllers, and/or parallel port controllers for connection to printer and so on); an expansion bus 114 for bus connection to external or internal peripheral devices (for example, the peripheral component interconnect (PCI) bus); and other system chips 116 (for example, graphics and sound devices). Examples of computers of this type are personal computers (PCs) and workstations. However, the application of the invention to other computing devices such as mainframes, embedded microcomputers in control systems, and personal digital assistants (PDAs) (in which case some of the indicated devices such as disk drive controllers may be absent) is also disclosed herein.

Pages 15-16, bridging paragraph:

Trap calls [[2012]] are added to the critical operating system, to detect states and request some actions in response. A trap call here means a call which causes the processor to save the current context (e.g. state of registers) and load a new context. Thus, where virtual memory addressing is used, the address pointers are changed.

Page 16, 3rd full paragraph:

Additional "virtual" drivers [[2014]] are added which, to the operating system, appear to provide access to an input/output (I/O) bus, allowing data to be written to the bus. In fact, the virtual bus driver [[2014]] uses memory as a communications medium; it exports some private memory (for input data) and imports memory exported by other systems (for output data). In this way, the operating system 201 (or an application running on the operating system) can pass data to another operating system (or application running on it) as if they were two operating systems running on separate machines connected by a real I/O bus.

Page 17, 2nd and 3rd paragraphs:

In step 310, the secondary operating system kernel 202 is modified to allow it to function in a multiple operating system environment, which is treated as a new

hardware architecture. As in step 306, the boot and initialisation sequences are modified, to allow the secondary operating system to be started by the hardware resource dispatcher, and to prevent it from accessing the hardware resources assigned to the other systems, as specified in the hardware resource dispatcher table. As in step 306, trap calls [[2022]] are added, to pass control to the hardware resource dispatcher.

Native drivers for shared system devices are replaced by new drivers [[2028]] dealing with devices which have been virtualized by the hardware resource dispatcher (interrupt controller, I/O bus bridges, the system timer and the real time clock). These drivers execute a call to virtual device handlers 416 of the hardware resource dispatcher in order to perform some operations on a respective device of the computer 100. Each such virtual device handler 416 of the hardware resource dispatcher is paired with a "peer" driver routine in the critical operating system, which is arranged to directly interact with the system device. Thus, a call to a virtual device handler is relayed up to a peer driver in the critical system for that virtualized device, in order to make real device access. As in step 306, read and write drivers [[2024]] for the virtual I/O bus are provided, to allow inter-operating system communications.

Page 18, 1st and 2nd paragraphs:

The interrupt service routines of the secondary operating system are modified, to provide virtual interrupt service routines [[2026]] each of which responds to a respective

virtual interrupt (in the form of a call issued by an interrupt handler routine 412 of the hardware resource dispatcher), and not to respond to real interrupts or events. In that way, the secondary operating systems 202, ... are therefore pre-emptable by the critical operating system 201; in other words, the secondary operating system response to a virtual interrupt can itself be interrupted by a real interrupt for the critical operating system 201. This typically includes:

- masking/unmasking events (interrupts at processor level);
- saving/restoring events mask status;
- identifying the interrupt source (interrupt controller devices);
- masking/unmasking interrupts at source level (interrupt controller devices).

New virtual device drivers [[2028]] are added, for accessing the shared hardware devices (the I/O bus bridges, the system console, the system timer and the real time clock). These drivers execute a call to virtual device handlers 416 of the hardware resource dispatcher in order to write data to, or read data from, a respective device of the computer 100.

Pages 18-19, bridging paragraph:

To effect this, the Linux kernel [[207]] is modified in this embodiment by adding new virtual hardware resource dispatcher architecture sub trees (nk-i386 and nk-ppc for

the I-386 and PowerPC variants) with a small number of modified files. Unchanged files were reused in their existing form. The original sub-trees were retained, but not used.

Page 19, lines 8-10:

- storing a table [[(403)]] which stores a list of hardware resources (devices such as ports) and an allocation entry indicating to which operating system each resource is uniquely assigned;

Page 29, line 16:

~~http://www.~~jaluna.com/developer/papers/CSI-TR-96-34.pdf.